WHAT IS CLAIMED IS:

1. A method for operating a computer system comprising:
receiving in the system a description of a finite state machine including a temporal logic condition; and
generating code for emulating the described finite state machine.

2. The method of claim 1, wherein:
the received description comprises at least two state definitions and at least one definition of a transition between states; and wherein
the received description comprises a conditional expression associated with a first state of the finite state machine, the conditional expression comprising a first temporal logic condition defined by a first temporal logic operator operating on an event, the conditional expression defining a logical condition for taking a first action specified in the description ; and wherein
generating code for emulating the described finite state machine comprises generating code for evaluating the conditional expression during emulation.

3. The method of claim 2, wherein generating code for evaluating the conditional expression comprises:
generating code for declaring a counter variable that is not otherwise specified in the description of the finite state machine;
generating code for initializing the counter variable upon entry into said first state;
generating code for incrementing the counter variable when said first event occurs;
generating code for performing a first test associated with said first temporal logic operator on the counter variable when said first state is active; and
generating code for taking a first specified action based on the result of said first test.

4. The method of claim 3, wherein the conditional expression is part of a conditional action expression in the definition of said first state, and wherein said first specified action is defined in the conditional action expression.

5. The method of claim 3, wherein the conditional expression is part of the definition of a transition from said first state to a second state and wherein said first specified action is defined by said transition.

6. The method of claim 3, wherein the description of the finite state machine further comprises a second conditional expression associated with a second state of the finite state machine, the second conditional expression comprising a second temporal logic condition defined by a second temporal logic operator operating on said event, the second conditional expression defining a logical condition for taking a second action specified in the description and wherein generating code for emulating the finite state machine further comprises:

generating code for initializing the counter variable upon entry into said second state;

generating code for performing a second test associated with said second temporal logic operator on the counter variable when said second state is active; and

generating code for taking a second specified action based on the result of said second test.

7. The method of claim 1, wherein the description of a finite state machine is a graphical description.

8. The method of claim 2, wherein said first temporal logic operator operates on an event E and a threshold T and is true when the event E has occurred at least T times during the current activation of said first state.

9. The method of claim 2, wherein said first temporal logic operator operates on an event E and a threshold T and is true when the event E has occurred at less than T times during the current activation of said first state.

10. The method of claim 2, wherein said first temporal logic operator operates on an event E and a threshold T and is true when the event E has occurred exactly T times during the current activation of said first state.

11. The method of claim 2, wherein said first temporal logic operator operates on an event E and a threshold T and is true when the event E has occurred a positive integral multiple of T times during the current activation of said first state.

12. The method of claim 7, wherein the graphical representation is a Stateflow® diagram.

13. The method of claim 7, wherein the conditional expression is part of a conditional action expression which is graphically represented as a textual expression within a node representing a state of the finite state machine.

14. The method of claim 7, wherein the conditional expression is part of the definition of a transition from said first state to a second state and the conditional expression is graphically represented as a textual expression that is proximate to a line connecting nodes representing the first and second states.

15. The method of claim 1, wherein the generated code is source code in human readable form.

16. A method for operating a computer system comprising:
receiving in the system a description of a finite state machine including a temporal logic condition; and
emulating the described finite state machine.

17. The method of claim 16, wherein
the received description comprises at least two state definitions and at least one definition of a transition between states; and wherein
the received description comprises a conditional expression associated with a first state of the finite state machine model, the conditional expression comprising a first temporal logic condition defined by a first temporal logic operator operating on an event, the conditional expression defining a logical condition for taking a first action specified in the model; and wherein

emulating the described finite state machine comprises evaluating the conditional expression during emulation.

18. The method of claim 17, wherein the emulating step further comprises:
allocating a counter variable that is not otherwise specified in the description of the finite state machine model;
initializing the counter variable upon entry into said first state;
incrementing the counter variable when said first event occurs;
performing a first test associated with said first temporal logic operator on the counter variable when said first state is active; and
taking a first specified action based on the result of said first test.

19. A computer programming system, comprising:
means for receiving in the system a description of a finite state machine including a temporal logic condition; and
means for generating code for emulating the described finite state machine.

20. A computer programming system comprising:
means for receiving in the system a description of a finite state machine including a temporal logic condition; and
means for emulating the described finite state machine.

21. A computer programming system, comprising:
a graphical user interface for receiving in the system a description of a finite state machine including a temporal logic condition; and
a code generator for generating code for emulating the finite state machine.

22. A computer programming system comprising:
a graphical user interface for receiving in the system a description of a finite state machine including a temporal logic condition; and
an interpreter for interpreting the received description to emulate the finite state machine.

23. A computer software product residing on a computer readable medium, the software product comprising instructions for causing a computer system to:
receive in the system a description of a finite state machine including a temporal logic condition; and
generate code for emulating the described finite state machine.

24. A computer software product residing on a computer readable medium, the software product comprising instructions for causing a computer system to:
receive in the system a description of a finite state machine including a temporal logic condition; and
emulate the described finite state machine.

25. A computer programming system comprising:
a central processing unit;
a mass storage subsystem;
a program editor capable of receiving from a user a description of a finite state machine including a temporal logic condition and storing the description on the mass storage subsystem;
a code generator capable of receiving the stored description and generating code for emulating the described finite state machine.

26. A computer programming system comprising:
a central processing unit;
a mass storage subsystem;
a program editor capable of receiving from a user a description of a finite state machine including a temporal logic condition and storing the description on the mass storage subsystem;
an emulator capable of receiving the stored description and emulating the described finite state machine.